Prediction Assignment - Coursera

Frederick Orndorff

12/9/2019

Introduction

It is now possible to collect a large amount of data about personal movement using activity monitoring devices such as a Fitbit, Nike Fuelband, or Jawbone Up. These type of devices are part of the "quantified self" movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. But these data remain under-utilized both because the raw data are hard to obtain and there is a lack of statistical methods and software for processing and interpreting the data.

This assignment makes use of data from a personal activity monitoring device. This device collects data at 5 minute intervals through out the day. The data consists of two months of data from an anonymous individual collected during the months of October and November, 2012 and include the number of steps taken in 5 minute intervals each day.

Data

The data for this assignment can be downloaded from the course web site:

- Dataset: Activity monitoring data
- Training Set
- Testing Set

The dataset is stored in a comma-separated-value (CSV) file and there are a total of 19,622 observations in the training set and the testing set includes 20 observations dataset.

```
#Load Packages and import data
library(caret); library(randomForest);
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.4.4
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart); library(rpart.plot)

#Make sure the working directory is where the *.csv files are located
```

```
training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
```

Data Processing (Cleaning)

Before we start the prediction we should ensure the data is not missing any values and is in the correct format.

```
training <- training [, colSums(is.na(training)) == 0]
testing <- testing [, colSums(is.na(testing)) == 0]
```

It looks like the first seven columns are not needed - I am going to delete and hopefully it speeds up the analysis. Also this might provide cleaner outputs.

```
training <- training [, -c(1:7)]
testing <- testing [, -c(1:7)]
```

Create a validation set

I am unsure if there is a need for a validation set - but this is what was taught in class, so the code below creates a validation set of 30 percent of the testing set.

```
set.seed(1111)
training_2 <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train_set <- training[training_2,]
validation_set <- training[-training_2,]
```

Prediction Models

Lets start with classification trees and random forests to predict the class of exercise.

Classification Trees

```
class_train <- trainControl(method = 'cv', number = 10)
fit_rpart <- train(classe ~., data = train_set, method = 'rpart')
print(fit_rpart, digits = 3)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa
```

```
##   0.0353  0.517     0.3727
##   0.0596  0.417     0.2113
##   0.1157  0.335     0.0797
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0353.
```

Lets see a plot:

```
library(rattle)
```

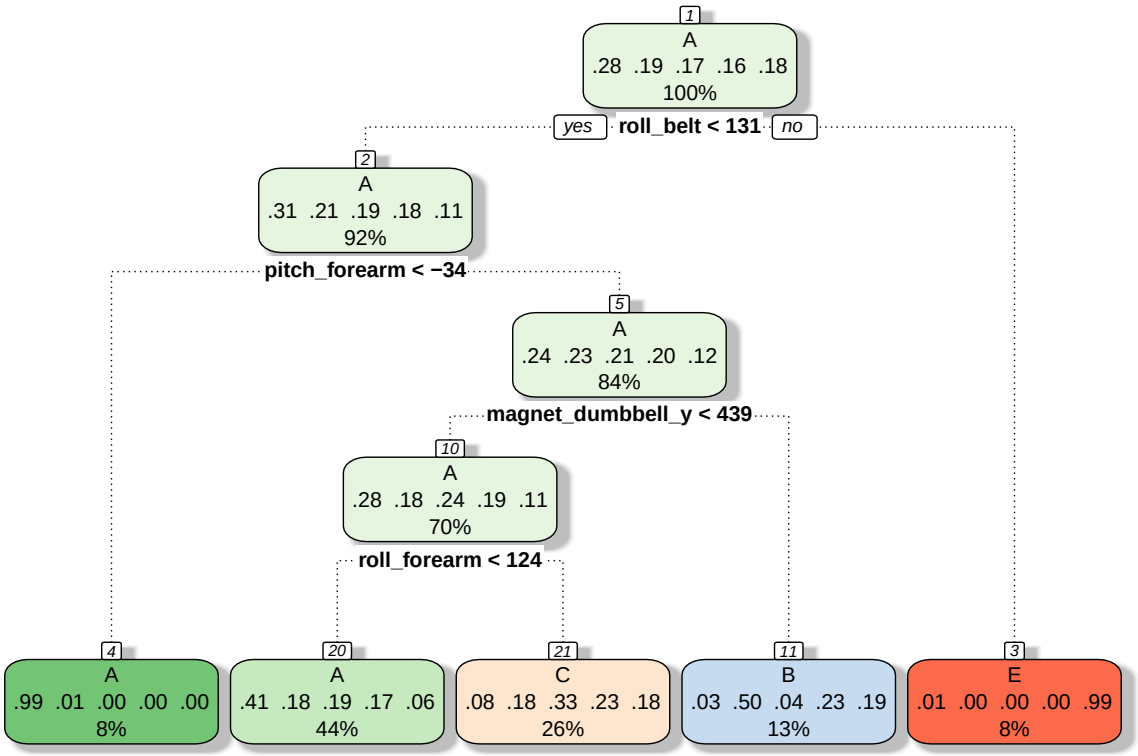
```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
fancyRpartPlot(fit_rpart$finalModel)
```



Rattle 2019−Dec−10 22:13:17 christinaorndorff

Confusion Matrix:

```
predict_rpart <- predict(fit_rpart, validation_set)
print (cm_rpart <- confusionMatrix(validation_set$classe, predict_rpart))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1524   27  120    0    3
##          B  478  386  275    0    0
##          C  458   32  536    0    0
##          D  447  159  358    0    0
##          E  162  141  296    0  483
##
## Overall Statistics
##
##                Accuracy : 0.4977
##                  95% CI : (0.4849, 0.5106)
##     No Information Rate : 0.5215
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.3434
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4966  0.51812  0.33817       NA  0.99383
## Specificity            0.9467  0.85350  0.88605   0.8362  0.88905
## Pos Pred Value         0.9104  0.33889  0.52242       NA  0.44640
## Neg Pred Value         0.6331  0.92436  0.78411       NA  0.99938
## Prevalence             0.5215  0.12659  0.26933   0.0000  0.08258
## Detection Rate         0.2590  0.06559  0.09108   0.0000  0.08207
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7217  0.68581  0.61211       NA  0.94144
```

```
print (acc_rpart <- cm_rpart$overall[1])
```

```
## Accuracy
## 0.497706
```

A 49.8% accuracy is pretty bad; we do better with Random Forests!!

Random Forests

```
#This takes too long
#fit_rf <- train(classe ~., data = train_set, method = 'parRF')
#lets try something else
fit_rf <- randomForest(classe ~., data = train_set)
print (fit_rf, digits = 3)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train_set)
##                Type of random forest: classification
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.57%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   16 2637    5    0    0 0.007900677
## C    0   17 2375    4    0 0.008764608
## D    0    0   23 2227    2 0.011101243
## E    0    0    3    4 2518 0.002772277
```

WOW - that took FOREVER to finish... maybe it is time to upgrade the computer??

Lets check the confusion matrix and the overall accuracy:

```
predict_rf <- predict(fit_rf, validation_set)
print (confusion_rf <- confusionMatrix(validation_set$classe, predict_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B    3 1134    2    0    0
##          C    0    5 1021    0    0
##          D    0    0   10  954    0
##          E    0    0    1    2 1079
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9930   0.9874   0.9979   1.0000
## Specificity            0.9993   0.9989   0.9990   0.9980   0.9994
## Pos Pred Value         0.9982   0.9956   0.9951   0.9896   0.9972
## Neg Pred Value         0.9993   0.9983   0.9973   0.9996   1.0000
## Prevalence             0.2845   0.1941   0.1757   0.1624   0.1833
## Detection Rate         0.2839   0.1927   0.1735   0.1621   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9987   0.9960   0.9932   0.9979   0.9997
```

```
print (accuracy_rf <- confusion_rf$overall[1])
```

```
## Accuracy
## 0.995582
```

It seems like the random forests provides a better prediction accuracy – but it cost me may minutes of surfing

the web – waiting for the computations to be completed. The last step is to test the model(s) on the test set.

Final Prediction

```
(predict(fit_rf, testing))
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```